


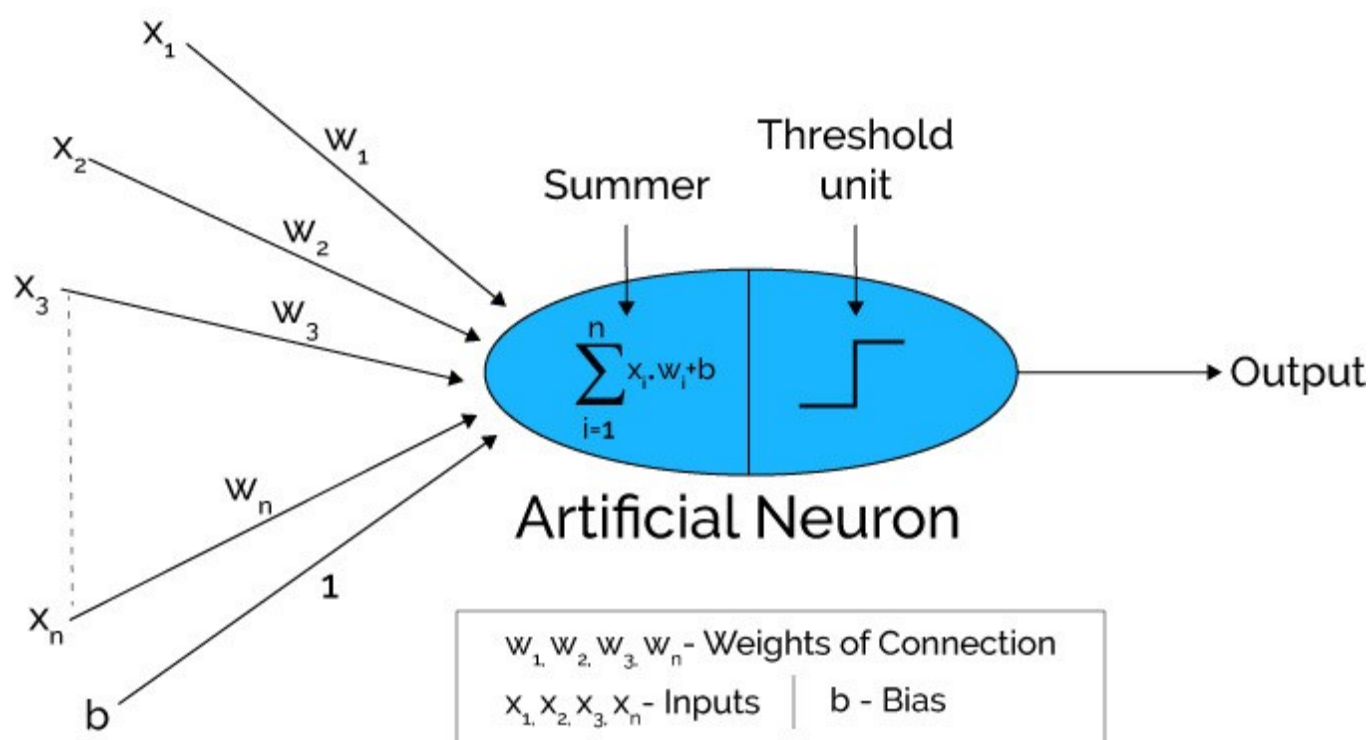
Name:dd

Time:2018.2.5

Title : BP神经网络原理研究

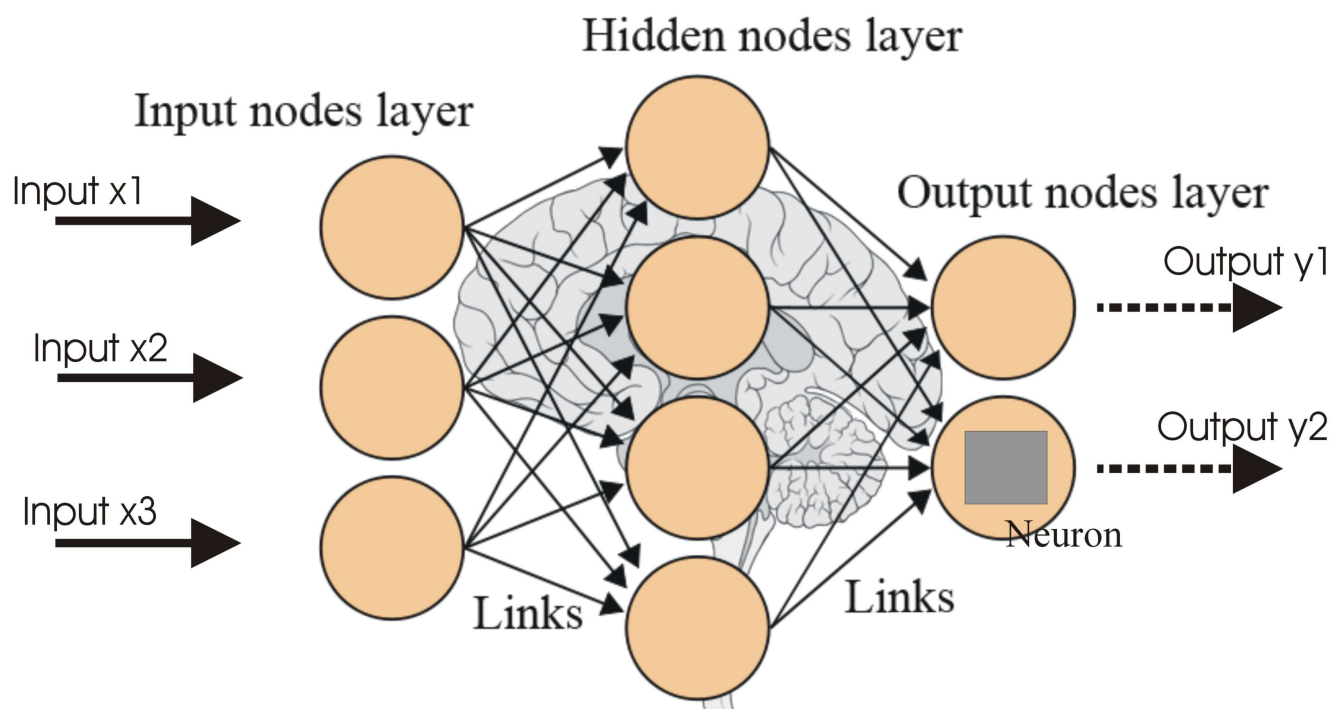


有一个单细胞生物，生活在糖水和盐水混合物中，糖水盐水在不停的流动，它有两个鞭毛，分别可以感知糖水和盐水，当总浓度到达一定值时，就会吃这些糖水盐水



但是，同时吃进糖水和盐水，会造成细胞结石，有损细胞健康

慢慢地，单细胞生物慢慢和其它细胞一起，组成了多细胞生物，进化出了一个能力，有糖水吃糖水，有盐水吃盐水，同时都有的话就不吃，防止细胞里面无法同时消化两种物体。



下面，我们用人工神经网络来模拟从单细胞到多细胞的进化过程

首先，画出最终的输入输出函数3维曲线。

In []:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)

x1=np.arange(0,2,1)
x2=np.arange(0,2,1)

X,Y=np.meshgrid(x1, x2)    # x-y 平面的网格
Z=np.logical_xor(X,Y)

ax.plot_surface(X, Y, Z, rstride = 1, cstride = 1, cmap = plt.get_cmap('rainbow'))
ax.contour(X, Y, Z, offset = -0.5, cmap = 'rainbow')
ax.set_zlim(-1, 1)

plt.show()

print(X)
print(Y)
print(Z)
```

In []:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = Axes3D(fig)

# X, Y value
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)    # x-y 平面的网格

R = np.sqrt(X ** 2 + Y ** 2)
# height value
Z= np.sin(R)
ax.plot_surface(X, Y, Z, rstride = 1, cstride = 1, cmap = plt.get_cmap('rainbow'))

# 绘制从3D曲面到底部的投影
#ax.contour(X, Y, Z, zdim = 'z', offset = -2, cmap = 'rainbow')

# 设置z轴的维度
ax.set_zlim(-2, 2)

plt.show()
```

用神经网络来学习XOR函数迭代版

In []:

```
import numpy as np  # Note: there is a typo on this line in the video
```

In []:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(-1,5,0.2)

y=x*x+np.sqrt(x+1)-3*x

y0=(x+x)*0

#print(x,y)

plt.plot(x,y)
plt.plot(x,y0)

plt.show()

#print("x,y,y1,yh")

x=0.5
#print(x)
y1=x*x+np.sqrt(x+1)-3*x
#print(y1)

x=2.1824
#print(x)
yh=x*x+np.sqrt(x+1)-3*x
#print(yh)
```

首先，定义一个sigmoid函数

In [280]:

```
def nonlin(x, deriv=False):
    if(deriv==True):
        return (x*(1-x))

    return 1/(1+np.exp(-x))
```

1/(1+exp(-x)) 导数推导

$(1/(1+\exp(-x)))' = ((1+\exp(-x))^{-1})' = (-1)((1+\exp(-x))^{-2})(\exp(-x))' = (-1)((1+\exp(-x))^{-2})(\exp(-x))'$ 而 $(\exp(-x))'$ 可以先转成 $(\exp(x)^{-1})'$ ，于是她又是一个复合函数的求导，即 $(\exp(x)^{-1})$ 对 $\exp(x)$ 的导数再乘上 $\exp(x)$ 对 x 的导数，又基本初等函数求导公式告诉我们， $(\exp(x))' = \exp(x)$ ，所以 $(\exp(-x))' = (\exp(x)^{-1})' = (-1)(\exp(x)^{-2})(\exp(x))' = (-1)(\exp(x)^{-2})\exp(x) = (-1)(\exp(x)^{-1}) = (-1)\exp(-x)$ 那么： $(-1)((1+\exp(-x))^{-2})(\exp(-x))' = (-1)((1+\exp(-x))^{-2})(-1)\exp(-x) = \exp(-x)((1+\exp(-x))^{-2}) = \exp(-x)/((1+\exp(-x))^2)$

$$f(\text{net}) = \frac{1}{1 + e^{-\text{net}}}$$

$$f'(\text{net}) = \frac{e^{-\text{net}}}{(1 + e^{-\text{net}})^2}$$

$$= \frac{1 + e^{-\text{net}} - 1}{(1 + e^{-\text{net}})^2}$$

$$= \frac{1}{1 + e^{-\text{net}}} - \frac{1}{(1 + e^{-\text{net}})^2}$$

$$= y(1 - y)$$

In []:

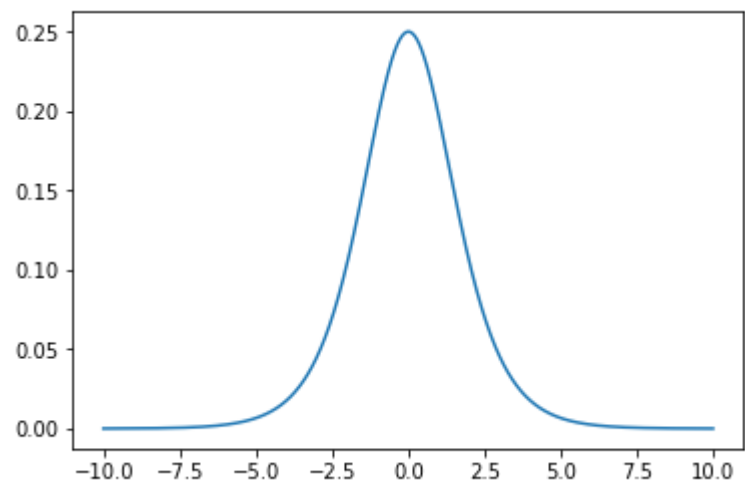
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(-10,10,2)
y=nonlin(x)
#print(y)
plt.plot(x,y)
plt.show()
```

In [282]:

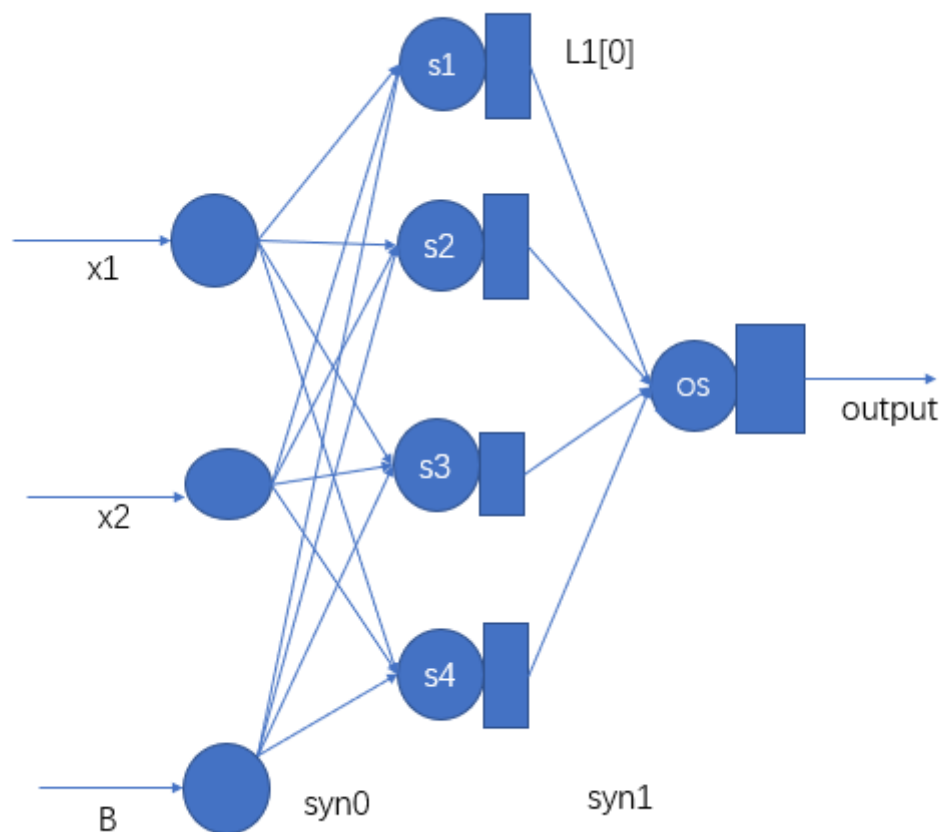
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(-10,10,0.01)
y=1/(1+np.exp(-x))
yderiv=nonlin(y,deriv=True)
plt.plot(x,yderiv)
plt.show()
```



复习一下表格试凑法

B	x1	x2	WB1	WB2	WB3	WB4	W11	W12	W13	W14	W21	W22	W23	W24	S1	S2	S3	S4	TS1	TS2	TS3	TS4	L2w1	L2W2	L2W3	L2W4	output
1	0	0	10	0.2	-1	-20	-10	-20	20	20	-10	20	-20	20	10	0.2	-1	-20	1	0	0	0	0	0	1	1	0
1	0	1	10	0.2	-1	-20	-10	-20	20	20	-10	20	-20	20	0	20	-21	0	0	1	0	0	0	0	1	1	0
1	1	0	10	0.2	-1	-20	-10	-20	20	20	-10	20	-20	20	0	-20	19	0	0	0	1	0	0	0	1	1	0
1	1	1	10	0.2	-1	-20	-10	-20	20	20	-10	20	-20	20	-10	0.2	-1	20	0	0	0	1	0	0	1	1	0
			WB1	WB2	WB3	WB4																					
			WB1	10	0.2	-1	-20																				
			W11	-10	-20	20	20																				
			W21	-10	20	-20	20																				
			W21	W22	W23	W24																					



In []:

```
syn0=np.array([[[-99,-99,99,99.0],
                [-99,99,-99,99],
                [99,0,0,99]]])
#print(syn0)

syn1=np.array([[0],
                [1.0],
                [1.0],
                [0]])
```

所有可能的输入值放在一个向量中

In [304]:

```
#input data
xinputs = np.array([[0,0,1],
                    [0,1,1],
                    [1,0,1],
                    [1,1,1]])
I0=xinputs
```

计算第一层的线性输出

In []:

```
S01=np.dot(xinputs, syn0)
#print(S01)
```

In []:

```
l1=nonlin(S01)
#print(l1)
```

In []:

```
S02=np.dot(l1, syn1)
#print(S02)
```

In []:

```
l2=nonlin(S02)
#print(l2)
```

希望的的输出结果

In [309]:

```
#output data
y = np.array([[0],
               [1],
               [1],
               [0]])
```

误差是

In []:

```
l2_error=y-l2
#print(l2_error)
```

为了直观地得出总误差多大，引入绝对值平均值lmabs

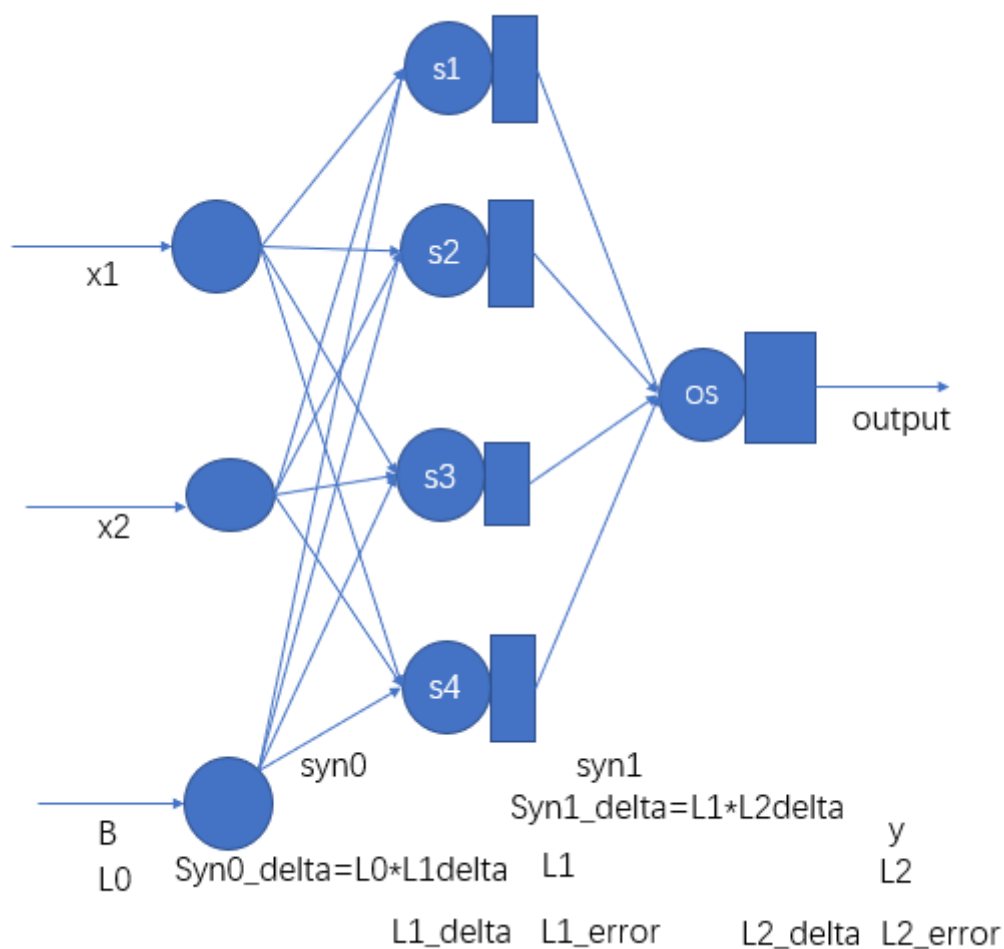
In []:

```
#误差的平均绝对为
lms=np.mean(np.abs(l2_error))
print(lms)

sk=[]
yerrorlms=[]
syn1_0=[]
sks=0

syn1_0.append(syn1[0,0])
sk.append(sks)
yerrorlms.append(lms)
sks=sks+1
```

误差反向传播



利用误差修正第二层权值网络

In [312]:

```
l2_delta = l2_error*nonlin(l2, deriv=True)
```

In []:

```
#print(l2_delta)
```

计算第一层的误差

In []:

```
#print(syn1)
```

In []:

```
#print(syn1.T)
```

In []:

```
l1_error = l2_delta.dot(syn1.T)  
#print(l1_error)
```

计算第一层的修正值

In []:

```
l1_delta = l1_error * nonlin(l1,deriv=True)  
#print(l1_delta)
```

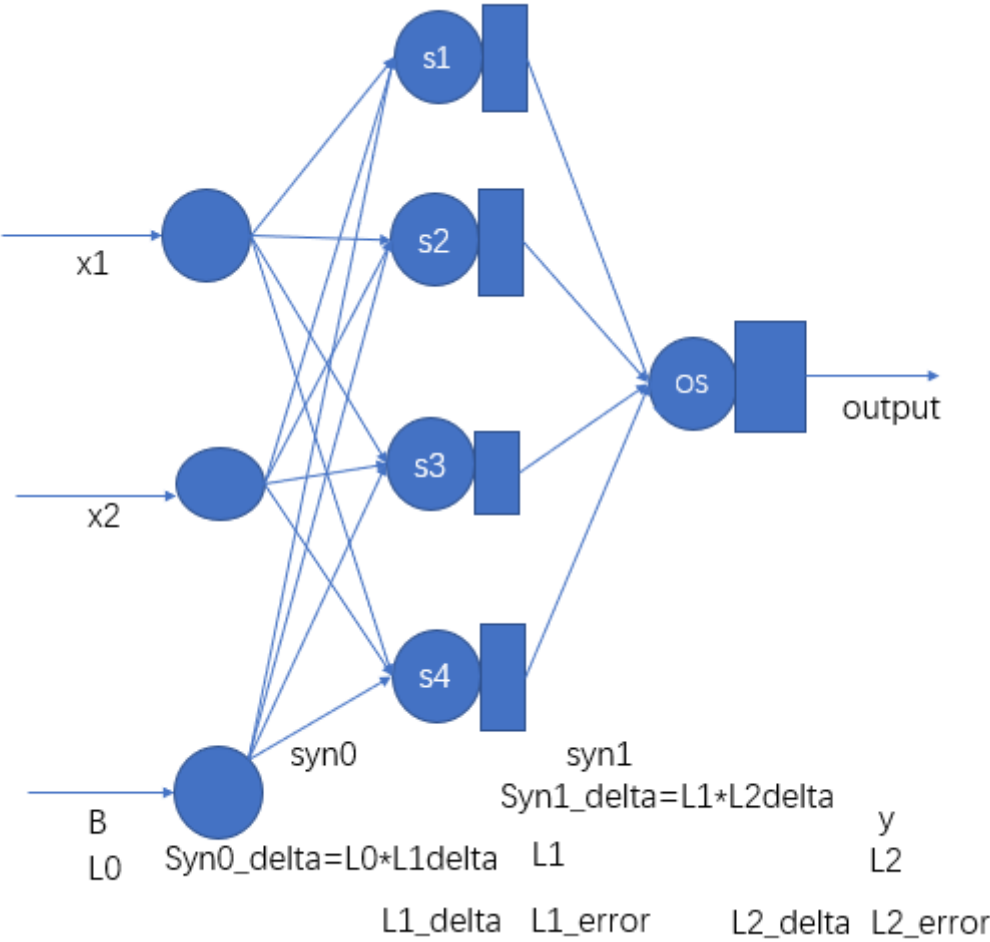
修正权值网络

In [318]:

```
syn1 += l1.T.dot(l2_delta)  
syn0 += l0.T.dot(l1_delta)
```

In []:

```
#print(syn0)  
#print(syn1)
```



机器迭代最终答案

In []:

```
for j in range(100):  
  
    #print("start forward calulate")  
    l0 = xinputs  
    #print("l0")  
    #print(l0)  
    l1 = nonlin(np.dot(l0, syn0))  
    #print("l1")  
    #print(l1)  
    l2 = nonlin(np.dot(l1, syn1))  
    #print("l2")  
    #print(l2)  
  
    # Back propagation of errors using the chain rule.  
    l2_error = y - l2  
  
    #print("l2")  
    #print(l2)  
  
    #print("l2_error")  
    #print(l2_error)  
  
    lms=np.mean(np.abs(l2_error))  
    synl_0.append(synl[0,0])
```

```
sk.append(sks)
yerrorlmss.append(lms)
sks=sks+1

#print("Error:")
#print(lms)

l2_delta = l2_error*nonlin(l2, deriv=True)

#print("nonline derive")
#print(nonlin(l2, deriv=True))

#print("l2_delta")
#print(l2_delta)

#    l1_error = l2_delta.dot(syn1.T)

#print("syn1.T")
#print(syn1.T)

#print("l1_error")
#print(l1_error)

l1_delta = l1_error * nonlin(l1,deriv=True)

#print("nonlin(l1, deriv=True) ")
#print(nonlin(l1, deriv=True))

#update weights (no learning rate term)
syn1 += l1.T.dot(l2_delta)
syn0 += l0.T.dot(l1_delta)

#print("syn0 syn1")

#print(syn0)
#print(syn1)

print("Output after training")
print(l2)

print(syn1_0)
print(yerrorlmss)
```

作业：

1.画出syn1[0] 的值在迭代过程中与Error的绝对值平均值 yerrorlmss的对应关系曲线。

2.画出迭代过程中syn1_0的 变化曲线

3.画出迭代过程中yerrorlmss的 变化曲线

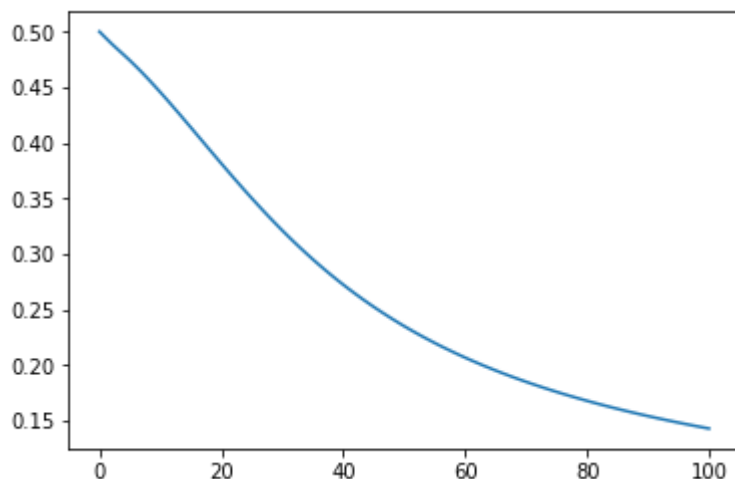
提示：

1. syn1[0]从0开始，到-4.8315
2. 声明list,然后append值,变量名syn1_0,yerrorlmss
3. 为了画出横坐标为迭代次数的图，需要创建对应顺序值，list命名为sk，变量名命名为sks，利用append添加每次的顺序值。
4. j循环体内以及j循环体前面的初始值均需要考虑

In [321]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x=sk
y=yerrorlmss
plt.plot(x, y)
plt.show()
print(x)
print(y)
```



```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 4
3, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 8
4, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
[0.5, 0.49448427379942766, 0.4890968219722739, 0.4840676097699067, 0.4791104063614
4164, 0.4740046148614543, 0.46868011003476656, 0.4631376690181409, 0.4574002447634
065, 0.45149451553847053, 0.44544547931530537, 0.4392756545828189, 0.4330055843264
5025, 0.42665448414009, 0.4202407044959697, 0.41378197236040015, 0.407295463838604
43, 0.4007977704604252, 0.3943048097861456, 0.3878317155610476, 0.3813927296829126
6, 0.3750011088332089, 0.3686690522629734, 0.36240765311770357, 0.3562268731310336
3, 0.35013553899701483, 0.34414135789054745, 0.33825094920963655, 0.33246988950724
393, 0.3268027676623294, 0.32125324754026785, 0.315824135664419, 0.31051745172939
9, 0.3053345001082855, 0.30027594082323894, 0.2953418587497121, 0.290531830100515
2, 0.2858449854827392, 0.2812800690356254, 0.27683549334060753, 0.272509389947047
1, 0.26829965548071605, 0.26420399339958645, 0.2602199515359941, 0.256344955618898
64, 0.2525763390078564, 0.24891136889432317, 0.24534726923868053, 0.24188124071525
252, 0.2385104779345943, 0.23523218420423053, 0.23204358407722459, 0.2289419339236
8523, 0.22592453074448599, 0.222988719429871, 0.22013189864879862, 0.2173515255382
8643, 0.21464511934598876, 0.21201026416395807, 0.20944461087719432, 0.20694587843
722717, 0.20451185455865623, 0.20214039592530034, 0.19982942798234798, 0.197576944
38162413, 0.19538100613873513, 0.19323974055337573, 0.19115133993739727, 0.1891140
601893001, 0.1871262192485453, 0.18518619545842663, 0.1832924258621446, 0.18144340
445311669, 0.17963768039739897, 0.17787385624332955, 0.17615058613109572, 0.174466
57401282303, 0.172820571891969, 0.17121137808922013, 0.16963783554073258, 0.168098
83013338048, 0.16659328908066723, 0.16512017934209233, 0.16367850608802717, 0.1622
6731121152949, 0.1608856718879933, 0.1595326991830832, 0.15820753670903057, 0.1569
093593290541, 0.15563737190941052, 0.15439080811837141, 0.15316892927124898, 0.151
97102322046058, 0.1507964032895115, 0.14964440724969844, 0.14851439633827201, 0.14
740575431675845, 0.14631788656811234, 0.14525021923136017, 0.1442021983723912, 0.1
4317328918955963]
```


画出迭代过程中syn1_0值的变化趋势

In []:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

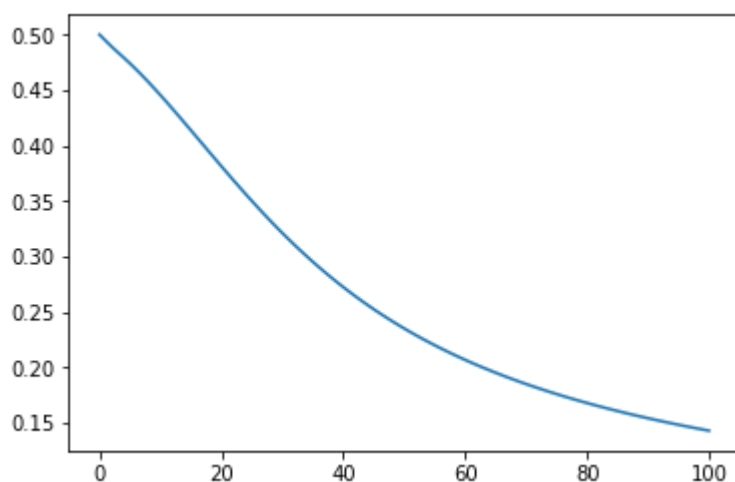
x=sk
y=yerrorlms
plt.plot(x, y)
plt.show()
```

画出迭代过程中，yerrorlms变化趋势

In [323]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
x=sk
y=yerrorlms

plt.plot(x, y)
plt.show()
```



作业：修改迭代次数，根据yerrorlms的趋势，确定迭代次数的最佳值